Release, Don't Wait!
Reliable Force Input Confirmation with Quick Release

Christian Corsten **Simon Voelker** **Jan Borchers**
RWTH Aachen University
52074 Aachen, Germany
{corsten, voelker, borchers}@cs.rwth-aachen.de

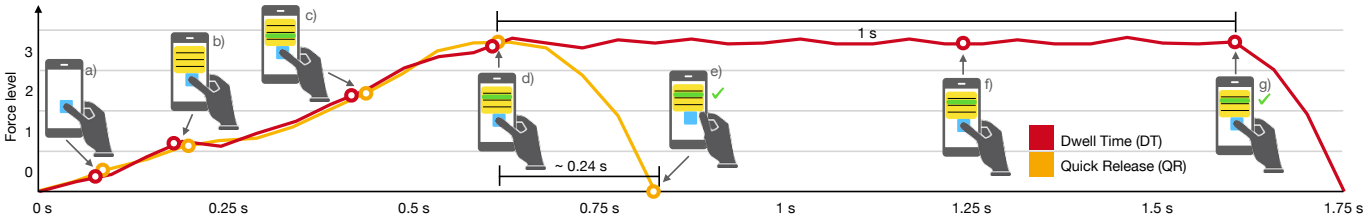

Figure 1. Interaction stages of force confirmation using QR and DT: In force level 0 (a) the user can normally interact with the system. By entering force level 1 (b), an on-screen menu is shown. Users can select a menu item by increasing the force to force level 2 (c) and force level 3 (d). Using QR, the selection is confirmed by quickly lifting the finger (e). Using DT, users have to maintain force (f) for a one second to confirm the selection (g).

ABSTRACT
Modern smartphones, like iPhone 7, feature touchscreens with co-located force sensing. This makes touch input more expressive, e.g., by enabling single-finger continuous zooming when coupling zoom levels to force intensity. Often, however, the user wants to select and confirm a particular force value, say, to lock a certain zoom level. The most common confirmation techniques are *Dwell Time* (DT) and *Quick Release* (QR). While DT has shown to be reliable, it slows the interaction, as the user must typically wait for 1 s before her selection is confirmed. Conversely, QR is fast but reported to be less reliable, although no reference reports how to actually detect and implement it. In this paper, we set out to challenge the low reliability of QR: We collected user data to (1) report how it can be implemented and (2) show that it is as reliable as DT (97.6% vs. 97.2% success). Since QR was also the faster technique and more preferred by users, we recommend it over DT for force confirmation on modern smartphones.

Author Keywords
Force; pressure; quick release; touchscreen; smartphone

ACM Classification Keywords
H.5.2 User Interfaces: Input devices and strategies (e.g., mouse, touchscreen)

INTRODUCTION
While force input on touchscreens has been researched as of the mid 80s [4], it recently has become an encouraging path to increase the richness of touch interaction also on mobile devices. Smartphones like iPhone 7 feature a touch screen with co-located force sensors to capture the normal force applied to the screen with each touch. This additional dimension for touch input can be exploited to control a parameter that is coupled to the continuously sampled force. For example, it enables single-finger zooming with the thumb when the device is used one-handed. In most cases, however, force is used to pick one value out of a range of continuous or discrete values. A recurring example from literature is menu control (e.g., [7, 8, 11, 18]). Each discrete menu item is mapped to a different range of adjacent force values. Depending on how much force the user applies, the appropriate item is selected.

The key problem, however, is to confirm the selection of an item. The most common techniques in the literature are *Dwell Time (DT)* and *Quick Release (QR)* [11]. Using DT, the selection is confirmed after maintaining force for a short duration (usually 1 s) for the corresponding item. Using QR, the selection is confirmed by quickly lifting the finger when the item is selected (Fig. 1). Although DT has shown to be a reliable force confirmation technique with $\approx 97\%$ success rates (e.g., [11, 15, 17]), it slows the interaction, as the user must typically wait before her selection is confirmed. In contrast, QR is a fast force confirmation technique, but it is less reliable than DT (e.g., $\geq$ 30% error rates in [6, 18]). Surprisingly, literature does *not* describe how QR is actually detected and implemented [11, 18].

In this paper, we challenged the low reliability of QR. We asked users to selected menu items by applying force with the thumb and to confirm each selection by quickly lifting the thumb off the mobile touchscreen. Based on this data we present how to implement QR. In a second study, we showed that users performed with this QR implementation as accurate as with DT (97.6% vs. 97.2% success). Since QR was also the faster force confirmation technique and more preferred by users, we recommended it over DT for force input confirmation on modern smartphones.

ISS '17, October 17—20, 2017, Brighton, United Kingdom

ISBN 978-1-4503-4691-7/17/10...$15.00
https://doi.org/10.1145/3132272.3134116

RELATED WORK

HCI research on force input dates back to the mid 80s. Buxton et al. [4] used continuous force sensing on a touch tablet to control the width of strokes drawn by the finger. Pens for graphic tablets [10, 11, 12] and force-sensitive mice [6, 13] have been explored to control desktop applications more efficiently. Force input has also been investigated on resistive mobile touchscreens [3, 18], e.g., to toggle small and capital letters during text input. More recently, smartphones and tablets with capacitive touchscreens have been equipped with force-sensing resistors at the bezel [8, 9, 17], sides [14, 16], or back [7] to let the fingers, that hold the device in place, partake in interactions, such as on-screen menu control. [1, 2] exploited users' difference in touch contact time and touch radius when applying various levels of force to the touchscreen to detect two levels of pseudo-force. However, their approach requires the user to always navigate with consistent timings, since longer touch contact time is interpreted as higher force. Various studies investigated the effect of feedback methods [11, 15, 18], transfer functions [6, 13, 15], and discrete force levels [8, 11, 15, 17, 18] on users' force input. In summary, users showed good performance with eight to ten discrete force levels on a 3–10 N range using a linear transfer function.

Another influencing factor is the force confirmation technique to confirm force input for a particular value. Most studies used *Dwell Time* (DT) [3, 6, 13, 15, 17] and *Quick Release* (QR) [3, 6, 18]. Both force confirmation techniques were first mentioned by Ramos et al. [11], who tested them for force input with a pen on a tablet. While DT has a low and consistent error rate ($\approx$ 3%) [3, 11, 15, 17], it slows the interaction, as the user must typically wait for 1 s before her selection is confirmed, resulting in $\approx$ 2.5 s completion times. Another drawback is that, once force is applied, DT does not allow the user to linger on an item longer than the dwell time. If she needs longer to decide which item to pick, then this leads to unintended confirmation [8]. In contrast, QR does allow for lingering and is fast ($\approx$ 1 s) [3, 11, 15, 17]. However, it has a high and inconsistent error rate ($\approx$ 5–40%) [3, 11, 15, 17], and is therefore often ranked lower than DT by users, although they prefer the faster completion time of QR [3, 11]. Current devices, like iPhone 7, neither use DT nor QR, but a simple thresholding technique: As soon as the user crosses a particular force value, she reaches the next menu state. This state is maintained as long as the user's force is lower than the next threshold. However, once a threshold is crossed, the user cannot go back, until the threshold is reset, e.g., by tapping on a back button.

Whereas the detection and implementation of DT for discrete force input can be clearly derived from its description, this is less clear for QR. Wilson et al. identified that "*Designing an accurate Quick Release mechanism is troublesome because it is difficult to identify a common and clear pattern of sensor behaviour from which user intent can be unambiguously retrieved.*" [18]. None of the studies using QR explained how they actually detected and implemented it. This might also explain the inconsistency in error rates, given different implementations. Due to this unclarity, we set out to investigate how to design a reliable QR mechanism by studying data from users performing the QR gesture.

DETECTING QUICK RELEASE

To get a rough estimate when a QR event happens, which is in line with finding out how long it takes the user to lift her finger off the touchscreen, we used the model human processor [5], also known as CMN model: According to this model, a user's action involves three steps, each of which takes a certain time: (1) perception (100 ms), (2) cognition (70 ms), (3) motion (70 ms). In the context of applying the QR gesture to confirm force input for a menu with discrete items, we obtain: (1) the user *perceives* which item is currently selected, (2) she then *decides* to confirm this item, and (3) she then *lifts* the finger off the touchscreen. Hence, a QR event should have happened $\approx$ 240 ms before the finger was lifted off the screen. However, since the CMN timings are only estimates and may differ by user and context, we conducted a controlled experiment to gather actual data from six participants (aged 24–29, M = 25.83, SD = 2.14, all male, all right-handed) performing the QR gesture. All participants were smartphone users but not experienced with force input.

Experimental Design

Users were asked to apply certain force on the touchscreen of a force-sensitive iPhone 6S to select a menu item and then confirm that item by quickly lifting the thumb of the screen. iPhone senses force values between 0 and $\frac{480}{72} \approx 6.67$ in steps of $\frac{1}{72}$ when force sensitivity is set to "firm". According to Apple's API documentation [1], a value close to 1.0 is equivalent to an ordinary touch, whereas any higher value indicates intentional force input. The documentation does not state how these values translate to Newtons, but experiments [7] hint at a $\approx$ 0–4 N range and a linear transfer function.

Fig. 2 shows the UI of the application that was used for displaying the task to the user and collecting the data. On the left, a vertical menu divided the device force range from 1.0–6.67 equidistantly into discrete segments. Force $<$ 1.0 was not visualized to let ordinary touch input coexist. The segments represented low to high force from bottom to top. When the force applied matched the range of a segment, it was filled with white color. A "+" indicated the segment that the user had to reach via pressing the thumb that should be placed at the location of the circled crosshair. Once the marked segment was reached, it turned green, and the user would immediately lift her thumb off the screen. Then, the next trial was shown.

Independent variables were MENU size (5, 10, and 15 segments), thumb LOCATION (Fig. 2), and force LEVEL (1.189, 2.418, 3.930, 5.347, and 6.481). Users did not have to reach these values exactly, but stay within the corresponding segment. This way, not all segments were tested, but it ensured that an equal amount of measurements per MENU was obtained—an approach also applied by [7, 8, 12, 17, 18]. The choice of LEVELS guaranteed that for each MENU, the lowest and the highest segment, a segment around the center,

[1] https://developer.apple.com/reference/uikit/uitouch

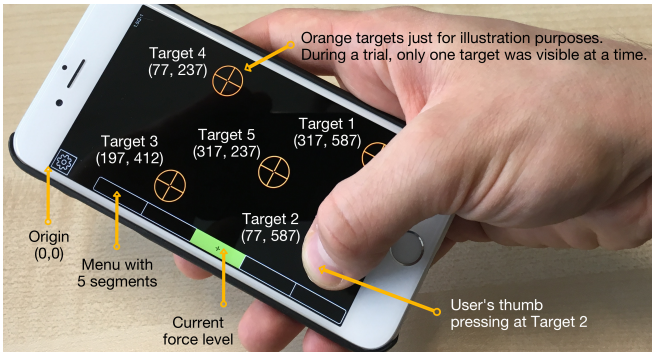

Figure 2. UI of the application used to collect data from users performing the QR gesture on a force-sensitive Phone 6S (667 × 375 pt screen).

a segment between the center and the lowest segment, and a segment between the center and the highest segment were chosen. Each combination from these variables was repeated three times, resulting in $3 \times 5 \times 5 \times 3 = 225$ trials per user.

MENU was counterbalanced using a Latin Square. LEVEL and LOCATION were combined and randomized. Once all three repetitions of the LEVEL × LOCATION trials were done, the user took a quick break and then continued with the next MENU. To become familiar with the task, each user first performed five test trials when MENU changed, resulting in $3 \times 5 = 15$ additional trials. The user was sitting in a chair without arm rests and held the phone in his right hand. A complete session took about 20 minutes per participant.

Dependent variables were *Force* [0–6.67], *Timestamp* [ms], *Frame* [i, $i \in \mathbb{N}_{\geq 1}$], and *Success* [0,1]. *Frame* denoted the i^{th} frame from the touch digitizer stream, which is sampled every $\approx$ 16 ms on iPhone. *Success* denoted whether the *Force* of a *Frame* was within the force range represented by the marked segment (1) or not (0).

Results

Since a QR event happens right before the end of a trial, we reversed our data, such that the *Timestamp* from the last *Frame* received from the digitizer (`touchesEnded` call in iOS) was set to 0 ms. This way, we could align all trials to the same final *Timestamp*, independent of how long the user needed to complete a trial. For convenience, we refer to discrete reversed *Frame*s in our analysis. Yet, multiplying *Frame* by $\approx$ 16 ms yields the equivalent continuous *Timestamp*.

Fig. 3a shows a typical plot of *Force* vs. reversed *Frame* for one user for MENU 5. The green color indicates that the user's *Force* (y-axis) matched the requested segment at the time the *Frame* (x-axis) was captured. The data shows that most matches are found around reversed *Frame* 16. This was generally independent from LEVEL and MENU: Fig. 3b shows the cumulated count of *Success* per reversed *Frame*. For each MENU there is a clear peak, and all three peaks almost overlap. If we use a *Frame*-to-*Force* lookup for the *Frame* at each peak and map the *Force* to the corresponding segment, we obtain promising success rates of 96% for MENU 5, 98% for MENU 10, and 95% for MENU 15. However, using just a single reversed *Frame* for the lookup might be problematic, especially when the user was jittering.

Therefore, we were interested in identifying a sequence of *Frame*s that lead to the highest success rate for each MENU SIZE. We applied the following optimization process: Assume that for one trial, $n \in \mathbb{N}_{>0}$ *Frames* were recorded. Then let F_i, $i \leq n$ be the i^{th} reversed *Frame* and let $w \in W = \{0...min(i-1, n-i)\}$ denote a width. Then $F_i(W) = \{F_i(w)\}$ is a set of sequences $F_{i-w}, \ldots, F_{i-1}, F_i, F_{i+1}, \ldots, F_{i+w}$ of $2w+1$ reversed *Frames* with F_i at the center. For each F_j in $F_i(w)$ $(1 \leq j \leq i+w)$ we calculated the menu segment $M(F_j)$ for the *Force* measured at that *Frame*. $Success(M(F_i(w))) \in \{0, 1\}$ denoted if the segment in $\{M(F_i(w))\}$ that occurred most frequently matched the requested segment for a trial.

Fig. 3c shows the success rates for MENU for $5 \leq i \leq 30$ and $0 \leq w \leq 30$. Although wider ranges for i and w could have been chosen, the peaks in Fig. 3b indicated that the optimum should be located within those ranges. As can be seen, there are multiple but few combinations of parameters i and w at the peaks of the curves. Although each MENU has a different F_i that leads to the maximum success rate (MENU 5: 96% for i =15, MENU 10: 98% for i =16, MENU 15: 97% for i =17) they share an optimal width of $w = 3$. Converting these sequences of $F_i(3)$ into time ranges, we obtain 192–288 ms for MENU 5, 208–304 ms for MENU 10, and 224–320 ms for MENU 15. As assumed, the 240 ms calculated from the CMN model are always contained within these ranges. As a next step, we wanted to see how users' performance for force confirmation with QR using these optimal parameters compares to confirmation with DT (Video Fig.).

VERIFICATION EXPERIMENT

We extended our first experiment to include QR as force confirmation technique using our previously determined parameters. The segment that occurred most frequently in $M(F_i(w))$ was feedbacked to the user as soon as she lifted her thumb. Half a second later, the next trial was shown. For baseline comparison, we also added DT: If the user maintained force for a segment for at least one second, it was confirmed through orange color. Lifting the thumb initiated the next trial. Twelve users (aged 23–53, *M* = 32.33, *SD* = 8.17, four female, all right-handed) participated. All participants were smartphone users but not experienced with force input, and none of them had participated in our previous experiment.

Independent variables from the first experiment were extended by TECHNIQUE, which denoted the two force confirmation techniques. Thumb LOCATIONs were slightly changed (Video Fig.) to test robustness against thumb placement. We did not change LEVEL since MENU 5 already included all five possible segments, and for MENU 10 and 15 a change of LEVEL would only have resulted in segments adjacent to the original ones. Counterbalancing and randomization were inherited; half of the users started with DT. Each participant performed 2 TECHNIQUE × 3 MENU × 5 LEVEL × 5 LOCATION × 3 repetitions = 450 trials. Before the next MENU was tested, the user performed five test trials, resulting in $2 \times 3 \times 5 = 30$ additional trials.

Dependent variables were *Success* $\in [0, 1]$ and *Time* [ms]. *Success* denoted whether the requested and the predicted seg-

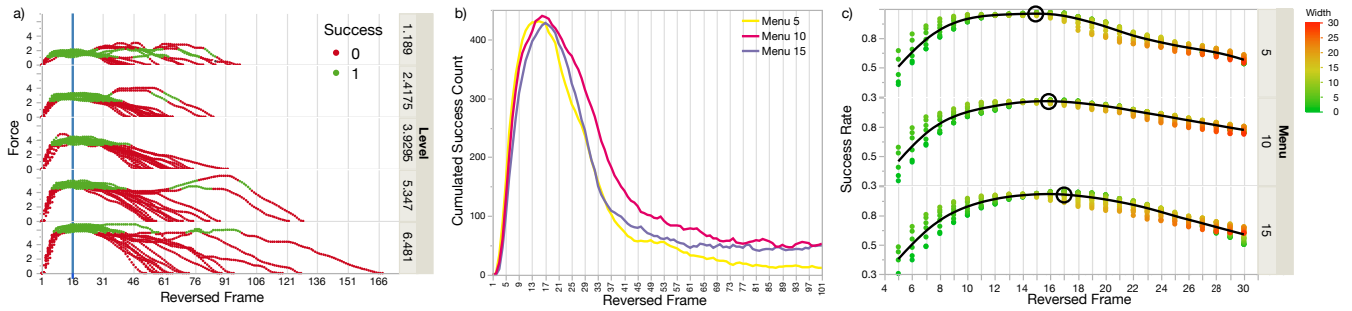

Figure 3. (a) Typical plot showing reversed *Frame* vs. *Force* data from one user for MENU 5. Around reversed *Frame* 16, the user's force always matched the requested LEVEL. Green data points from higher reversed *Frame*s indicate that users reached the correct LEVEL, but did not release yet, e.g., because they were jittering. (b) The cumulated count of *Success* confirms that this finding is independent from MENU and LEVEL. (c) Different combinations of reversed *Frame* and *width* lead to different *Success* rates. Circles indicate reversed *Frame*s for MENU that led to optimal *Success* with a *width* of $w = 3$.

ment matched (1) or not (0), and *Time* was measured until a trial was completed, i.e., until the dwell time expired or, for QR, when the thumb was no longer in contact with the touch digitizer. At the end, users were asked to vote for their preferred force confirmation technique, or whether they had no preference at all.

Results

Since we interested in a direct comparison between DT and QR, we always directly contrast both TECHNIQUEs for each variable. *Time* was log-transformed for repeated-measures ANOVAs. For *Success*, we conducted McNemar and Cochran's Q tests, since the data was dichotomous.

TECHNIQUE had a significant effect on ***Time*** ($F_{1,5377} = 348.60$, $p < .0001$): Using DT, users needed 2341 ms (95% CI: $\pm$51 ms) to complete a trial on average, which was twice as slow compared to 1125 ms (95% CI: $\pm$26 ms) for QR.

MENU had a significant effect on *Time* for both DT ($F_{2,2686} = 159.02$, $p < .0001$) and QR ($F_{4,2686} = 179.80$, $p < .0001$). Tukey HSD posthoc pairwise comparisons for each TECHNIQUE were all significant, i.e., *Time* increased with MENU for both, DT and QR. Pairwise t-tests of *Time* between the same MENUs across both TECHNIQUEs were all three significant ($F_{4,1787} \geq 1121.34$, adjusted $p < .001$, each), i.e., for each MENU, confirmation with QR was significantly faster (Fig. 4).

LEVEL had a significant effect on *Time* for both DT ($F_{4,2684} = 160.78$, $p < .0001$) and QR ($F_{4,2684} = 158.84$, $p < .0001$). For DT, Tukey HSD posthoc pairwise comparisons were not significant between the two lowest LEVELs, but between all other pairs ($p < .0001$, each). For QR, Tukey HSD posthoc pairwise comparisons were also not significant between the two lowest LEVELs and between the lowest and the highest LEVEL—for these, users were fastest—, but between all other pairs ($p < .0001$, each) (Fig. 4). Being faster at lower LEVELs is plausible since applying more force takes more time, and for the highest LEVEL, users could quickly apply as many force as they could to always land on the segment for the highest force level. Pairwise t-tests for *Time* between the same LEVELs across both TECHNIQUEs were all significant ($F_{1,1067} \geq 626.80$, adjusted p < 0.001, each). For any LEVEL, force confirmation with QR was always significant faster than with DT.

LOCATION had a significant effect on *Time* for DT ($F_{4,2684} = 9.03$, $p < .0001$): Tukey HSD pairwise comparisons showed that confirming force input with the thumb at the lower left corner of the screen was significantly slower than any other LOCATION ($p < .001$, each). For QR, however, LOCATION had no effect on *Time* ($F_{4,2684} = .97$, n.s.). Pairwise t-tests for *Time* between the same LOCATIONs across both TECHNIQUEs were all significant ($F_{1,1067} \geq 656.03$, adjusted p < 0.001, each). For each LOCATION, confirmation with QR was significantly faster (Fig. 4).

TECHNIQUE had no significant effect on ***Success*** ($\chi^2(1) = .35$, n.s.), i.e., we could not prove that any TECHNIQUE performed better than the other as regards *Success*.

MENU had a significant effect on *Success* for DT ($Q(2) = 20.60$, $p < .0001$). Posthoc pairwise comparisons revealed that *Success* for MENU 5 (99.2%) was significantly higher compared to MENUs 10 (96.6%) and 15 (96.0%) ($p < .001$, each). For QR, however, MENU had no effect on *Success* ($Q(2) = 2.49$, n.s.)—users performed equally well independent from the menu size. Pairwise comparisons of *Success* between the same MENUs across both TECHNIQUEs were significant for MENU 5 ($\chi^2(1) = 4.65$, $p < .05$) and for MENU 15 ($\chi^2(1) = 4.66$, $p < .05$): Whereas *Success* for MENU 5 was significantly higher for DT (99.2%) than for QR (97.9%), *Success* for QR was significantly higher for MENU 15 (97.9%) compared to DT (96.0%). Yet, these differences were smaller than 2% (Fig. 5).

LEVEL had a significant effect on *Success* for DT ($Q(4) = 28.23$, $p < .0001$). Posthoc pairwise comparisons revealed that *Success* for LEVEL 5.347 was significantly lower compared to all other LEVELs except for LEVEL 3.925, for which, however, *Success* was significantly lower compared to LEVEL 6.481 ($p < .05$, each). LEVEL also had a significant effect on *Success* for QR ($Q(4) = 24.38$, $p < .0001$). Pairwise McNemar tests showed that *Success* for LEVEL 6.481 was significantly higher compared to all other LEVELs except for LEVEL 2.4175 ($p < .001$, each). Pairwise comparisons of *Success* between the same LEVELs across both TECHNIQUEs were all not significant (Fig. 5).

Level		1.189	2.4175	3.9295	5.347	6.481
Mean	DT	2,228	2,189	2,623	3,071	1,591
	QR	956	973	1,288	1,540	869
95% CI ±	DT	107	105	108	136	48
	QR	49	42	64	74	37
Target		*1*	*2*	*3*	*4*	*5*
Mean	DT	2,663	2,248	2,325	2,189	2,279
	QR	1,138	1,088	1,130	1,139	1,132
95% CI ±	DT	140	97	113	93	109
	QR	62	54	59	61	57

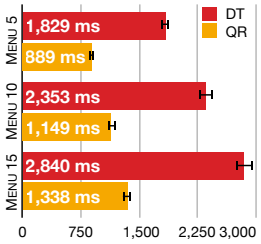

Figure 4. DT vs. QR results for *Time* in ms. Error bars denote 95%CI.

Level		1.189	2.4175	3.9295	5.347	6.481
Mean	DT	97.41	97.59	96.85	94.63	99.81
	QR	96.48	98.70	96.11	96.67	99.81
95% CI ±	DT	1.35	1.29	1.48	1.91	.36
	QR	1.56	.95	1.63	1.52	.35
Target		*1*	*2*	*3*	*4*	*5*
Mean	DT	98.33	97.96	96.67	96.49	96.85
	QR	97.04	97.78	97.04	97.78	98.15
95% CI ±	DT	1.08	1.19	1.52	1.57	1.48
	QR	1.44	1.25	1.44	1.25	1.14

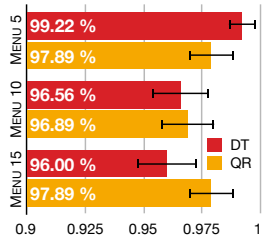

Figure 5. DT vs. QR results for *Success* in %. Error bars denote 95%CI.

Location neither had a significant effect on *Success* for DT ($Q(4) = 5.81$, n.s.) nor for QR ($Q(4) = 2.25$, n.s.). Likewise, pairwise comparisons of *Success* between the same Locations across both Techniques were also all not significant (Fig. 5).

Users' ***preference*** for Technique was significant ($Q(2) = 9.50$, $p < .001$): Nine users voted significantly higher for QR compared to two votes for DT and one for 'no preference' ($p < .05$, each).

Discussion

Not surprisingly, users were faster with QR than with DT; results for *Time* were in line to findings from related work. Interestingly, using DT took 1216 ms longer—216 ms more than the added DT. This could be explained by a strategy from users, who increased or decreased force very slowly when they jittered around a targeted segment. Also note that *Time* for DT stopped right after the timer expired, but in practice, the user would still need to lift her finger to continue interaction. This would add another 240 ms from the CMN model, which is, on the contrary, already included in *Time* for QR. Although Location did not influence *Success*, users were slower with DT when their thumb was placed at the lower right corner of the touchscreen. Maintaining force at that location is difficult, as the device is imbalanced and the thumb is folded. The QR gesture, however, did not involve maintaining force, which might explain why users were equally fast at any Location. *Success* for DT was in line with findings from related work. Yet, we could not find an overall difference between DT and our implementation of QR. What is more, is that unlike DT, QR was not influenced by Menu, which makes it more flexible as regards the choice of menu size. Combined with users' faster interaction time and higher preference for QR, we can summarize it as the more advantageous force confirmation technique compared to DT. We envision QR to allow for effective control of context menus, e.g., to quickly access shortcuts in smartphone applications.

LIMITATIONS AND FUTURE WORK

We collected all data on iPhone 6S. However, we expect similar results for other devices, since we saw that the QR gesture is in line with predictions from the device-independent CMN model. We also plan to optimize feedback visualization for QR: As of now, when the user lifts her thumb, the menu cursor will rapidly drop and then jump back to the confirmed segment. This is since we cannot start calculating which segment the user wanted to confirm before she completely lifted her thumb. A solution could be pausing visualization updates (Video Fig.) when a rapid decrease in force ($\delta_{Force} \leq$ -0.3 units on the 0–6.67 scale) for QR is detected, but this would probably be dependent from the sensor range and menu items.

CONCLUSION

In this paper we set out to challenge the low reliability of Quick Release (QR), a common technique to confirm force input by quickly lifting the finger. We contrasted it with Dwell Time (DT), a technique that requires the user to maintain pressure for 1 s to confirm the input, and that is reported as more reliable than QR. While detecting and implementing DT is clear and straightforward, literature does not describe how to do so for QR. Inspired by the CMN model [5], we hypothesized that the force the user intends to confirm can be retrieved by looking $\approx$ 240 ms back in time once the finger has been lifted. To confirm our hypothesis, we collected data from users who controlled menu items on a force-sensitive smartphone by pressing with the thumb. Based on this data set, we implemented an algorithm to detect QR: Use a force-to-menu item lookup between $\approx$ 200–300 ms before the finger is lift off the screen and pick the item that occurred most frequently within that time frame. In a verification study, we tested this implementation against DT: With a 97.6% success rate, QR was as reliable as DT, that had a 97.2% success rate. Combined with users' faster performance and higher preference for QR, we recommend it over DT as confirmation technique for force input on modern smartphones.

ACKNOWLEDGMENTS

This work was funded by the German B-IT Foundation. We thank Florian Busch for helping with the study setup and Oliver Nowak for editing the video figure.

REFERENCES

1. Arif, A. S., Mazalek, A., and Stuerzlinger, W. The Use of Pseudo Pressure in Authenticating Smartphone Users. In *Proc. MOBIQUITOUS '14*, ICST (2014), 151–160.
2. Arif, A. S., and Stuerzlinger, W. Pseudo-Pressure Detection and Its Use in Predictive Text Entry on Touchscreens. In *Proc. OzCHI '13*, ACM (2013), 383–392.
3. Brewster, S. A., and Hughes, M. Pressure-Based Text Entry for Mobile Devices. In *Proc. MobileHCI '09*, ACM (2009), 9:1–9:4.
4. Buxton, W., Hill, R., and Rowley, P. Issues and Techniques in Touch-Sensitive Tablet Input. In *Proc. SIGGRAPH '85*, ACM (1985), 215–224.
5. Card, S., and Moran, T. Newell (1983) The Psychology of Human-Computer Interaction. *The psychology of human-computer interaction. CRC*, 488.

6. Cechanowicz, J., Irani, P., and Subramanian, S. Augmenting the Mouse with Pressure Sensitive Input. In *Proc. CHI '07*, ACM (2007), 1385–1394.
7. Corsten, C., Daehlmann, B., Voelker, S., and Borchers, J. BackXPress: Using Back-of-Device Finger Pressure to Augment Touchscreen Input on Smartphones. In *Proc. CHI '17*, ACM (2017), 4654–4666.
8. McLachlan, R., Boland, D., and Brewster, S. Transient and Transitional States: Pressure as an Auxiliary Input Modality for Bimanual Interaction. In *Proc. CHI '14*, ACM (2014), 401–410.
9. McLachlan, R., and Brewster, S. Bimanual Input for Tablet Devices with Pressure and Multi-Touch Gestures. In *Proc. MobileHCI '15*, ACM (2015), 547–556.
10. Mizobuchi, S., Terasaki, S., Keski-Jaskari, T., Nousiainen, J., Ryynanen, M., and Silfverberg, M. Making an Impression: Force-Controlled Pen Input for Handheld Devices. In *Proc. CHI '12*, ACM (2005), 1661–1664.
11. Ramos, G., Boulos, M., and Balakrishnan, R. Pressure Widgets. In *Proc. CHI '04*, ACM (2004), 487–494.
12. Ramos, G. A., and Balakrishnan, R. Pressure Marks. In *Proc. CHI '07*, ACM (2007), 1375–1384.
13. Shi, K., Irani, P., Gustafson, S., and Subramanian, S. PressureFish: A Method to Improve Control of Discrete Pressure-based Input. In *Proc. CHI '08*, ACM (2008), 1295–1298.
14. Spelmezan, D., Appert, C., Chapuis, O., and Pietriga, E. Side Pressure for Bidirectional Navigation on Small Devices. In *Proc. MobileHCI '13*, ACM (2013), 11–20.
15. Stewart, C., Rohs, M., Kratz, S., and Essl, G. Characteristics of Pressure-Based Input for Mobile Devices. In *Proc. CHI '10*, ACM (2010), 801–810.
16. Wilson, G., Brewster, S., and Halvey, M. Towards Utilising One-handed Multi-Digit Pressure Input. In *Proc. CHI EA '13*, ACM (2013), 1317–1322.
17. Wilson, G., Brewster, S. A., Halvey, M., Crossan, A., and Stewart, C. The Effects of Walking, Feedback and Control Method on Pressure-Based Interaction. In *Proc. MobileHCI '11*, ACM (2011), 147–156.
18. Wilson, G., Stewart, C., and Brewster, S. A. Pressure-Based Menu Selection for Mobile Devices. In *Proc. MobileHCI '10*, ACM (2010), 181–190.